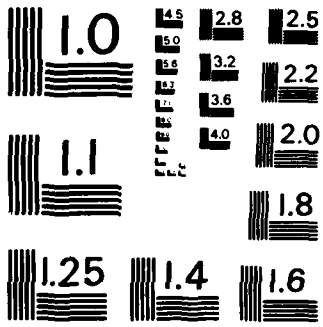
AD-A161 304 EXTENSION OF THE PARALLEL NESTED DISSECTION ALGORITHM TO THE PATH ALGEBRA PROBLEMS(U) HARVARD UNIV CAMBRIDGE MA AIKEN COMPUTATION LAB V PAN ET AL. 1985 TR-15-85 1/1

UNCLASSIFIED N00014-80-C-0647 F/G 12/1 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A161 304

(12)

EXTENSION OF THE PARALLEL NESTED DISSECTION ALGORITHM TO THE PATH ALGEBRA PROBLEMS

Victor Pan
and
John H. Reif

TR-15-85

Harvard University

Center for Research in Computing Technology

11 18 85 014

Aiken Computation Laboratory
33 Oxford Street
Cambridge, Massachusetts 02138

12

EXTENSION OF THE PARALLEL NESTED DISSECTION ALGORITHM TO THE PATH ALGEBRA PROBLEMS

Victor Pan
and
John H. Reif

TR-15-85

unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. AD A161304
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) EXTENSION OF THE PARALLEL NESTED DISSECTION ALGORITHM TO THE PATH ALGEBRA PROBLEMS	5. TYPE OF REPORT & PERIOD COVERED Technical Report
	6. PERFORMING ORG. REPORT NUMBER TR-15-85
7. AUTHOR(s) Victor Pan, John H. Reif	8. CONTRACT OR GRANT NUMBER(s) N00014-80-G0647
9. PERFORMING ORGANIZATION NAME AND ADDRESS Harvard University, Cambridge, MA 02138	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research, 800 North Quincy Street, Arlington, VA 22217	12. REPORT DATE 1985
	13. NUMBER OF PAGES 8
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same as above	15. SECURITY CLASS. (of this report)
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

nested dissection, parallel algorithms, grid graphs, planar graphs, minimum cost paths, graph connectivity, path problems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See reverse side.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Summary

The authors' recent parallel nested dissection algorithm for solving linear systems is extended in order to substantially accelerate several path algebra computations in both cases of a single source path and of all pair paths where the path problem is defined by a sparse matrix whose associated graph has a family of small separators.

- 1 -

Extension of the Parallel Nested Dissection Algorithm to the Path Algebra Problems.

Victor Pan *

Computer Science Department
State University of New York at Albany
Albany, New York
and

John Reif **

Aiken Computation Lab.
Division of Applied Sciences, Harvard University
Cambridge, MA

Summary

The authors' recent parallel nested dissection algorithm for solving linear systems is extended in order to substantially accelerate several path algebra computations in both cases of a single source path and of all pair paths where the path problem is defined by a sparse matrix whose associated graph has a family of small separators. Keyword: computations. (—

* Supported by NSF Grant MCS 8203232 and DCR-8507573.

** This work was supported by Office of Naval Research Contract N00014-80-C-0647.

Path algebra computations are required for the solution of numerous problems of practical interest, see [GM], [T]. In particular M. Gondran and M.Minoux list the applications of path algebras to the problems of: vehicle routing, investment and stock control, dynamic programming with discrete states and discrete time, network optimization, artificial intelligence and pattern recognition, labyrinths and mathematical games, encoding and decoding of information, [GM], pp. 41-42, 75-81. There is an effective way to unify such computations by reducing them to certain matrix operations over a *dioid* $(S,\oplus,\ast)$ where $\oplus$ and $\ast$ are the operations of that dioid; see [GM], pp. 84-102. The following classes of problems allow that reduction: i) existence (problems of connectivity); ii) enumeration (elementary paths, multicriteria problems, generation of regular languages); iii) optimization (paths of maximum capacity, paths with minimum number of arcs, shortest paths, longest paths, path of maximum reliability, reliability of a network); iv) *counting* (counting of paths, Markov chains); v) optimization and post-optimization (problems of k-th path, n-optimal paths), see [GM], pp. 91, 94-102.

Specifically, the above computations are reduced to the evaluation (over the dioid) of the matrix A^* (the all pair path problems) or of the vector $A^* \ast \mathbf{b}$ (the single source path problems) where

$$A^* = A^{(n-1)},\ A^{(q+1)} = A^{(q)} \oplus A^{q+1},\ q=0,1,\ldots,\ A^{(q+1)} = A^{(q)} \text{ if } q \geq n-1. \qquad (1)$$

$A^{(0)} = I$ is the identity matrix, A is an $n \times n$ input matrix, $\mathbf{b}$ is a fixed coordinate vector of dimension n, [GM], sect. 3.2, 3.3. Here and hereafter we assume that all computations, in particular computing the powers of a matrix, are performed over the dioid. Then

$$A^* = I \oplus A \oplus A^2 \oplus \cdots \oplus A^{n-1}$$

so A^* can be computed as follows,

$$A^* = (I \oplus A) \ast (I \oplus A^2) \ast (I \oplus A^4) \ast \cdots \ast (I \oplus A^{2^k}),\ k = \lceil \log_2 n \rceil .$$

This requires only k-1 matrix additions and 2k-2 matrix multiplications, which means a total of $n^2(k-1)(4n-1)$ operations in the dioid. (We cannot use fast matrix multiplication algorithms over the dioid.) For many dioids the operation $\oplus$ is idempotent, that is, $a \oplus a = a$ for all $a \in S$. In that case

$$A^* = \sum_{r=0}^{n} A^r = \sum_{r=0}^{n} C(n,r)\, A^r = (I \oplus A)^n = (I \oplus A)^{2^k}$$

(where $\sum$ denotes a sum in the dioid, $C(n,r) = r! \,/\, n!(n-r)!$, $k = \lceil \log_2 n \rceil$), so A^* can be computed via repeated squaring of $I \oplus A$. Therefore we may compute A^* using only k-1 matrix multiplications and a single addition of the two matrices A and I, that is, using a total of $n^2(k-1)(2n-1)+n$ operations in a dioid with idempotent $\oplus$. It is easy to parallelize these two known algorithms, which yields rather efficient parallel scheme for the evaluation of A^* where A is a dense matrix. If A is sparse, then the above ways are relatively less effective for the sparsity of A is not generally preserved during the computation.

Computing $A^* \ast b$ (the single source path problems), can be reduced to the n successive premultiplications of the vectors $\sum_{r=0}^{k} A^r b$ by the matrix A for $k=0,1,\ldots,n-1$ (and to n-1 vector additions in the case where the operation $\oplus$ is not idempotent in the given dioid), that is, to a total of $(2\,D(A) - n)(n-1)$ operations in the dioid (or of $2\,D(A)(n-1)$ operations in the case of nonidempotent $\oplus$), provided that the operations in the dioid are not counted if at least one of the operands is zero. Here D(A) denotes the number of nonzero entries of A. This way we exploit sparsity of A to some extent; note, however, that the above estimates translate into O(n log n) parallel steps and D(A) processors. Here and hereafter we assume a customary machine model of parallel computation, where in every parallel step each processor performs at most one operation of the considered type; in our case this means at most one operation of the dioid.

In this paper we will consider the case where the associated graph $G = (V,E)$ of the input matrix has an $s(n)$-separator family, $s(n) = n^\sigma$, $\sigma < 1$, σ is sufficiently small. We define an $s(n)$-separator family of a graph following Definition 1.1 of [PR], that is, G is said to have an $s(n)$-separator family if, by deleting a separator set S of vertices, $|S| \leq s(|V|)$, we may partition G into two disconnected subgraphs with the vertex sets V_1 and V_2 such that $|V_i| \leq \alpha |V|$, i=1,2, α is a constant, $\alpha < 1$; furthermore we assume that such a partition can be recursively extended to each of the two resulting subgraphs of G defined by the vertex sets $S \cup V_i$, i=1,2, and so on. Then we may further reduce the computational cost of computing A^* and $A^* \ast \mathbf{b}$ using the nested dissection algorithms of [LRT] for the sequential computation and of [PR] for the parallel computation.

It may seem that we need to have a symmetric positive definite matrix A to apply these algorithms. A is indeed symmetric in the case of paths in graphs; however, even when we deal with digraphs and nonsymmetric A, we may apply the extension of the algorithms of [PR] to the nonsymmetric case following [PRa], for instance, we may reduce the solution of the matrix equation $A\mathbf{x} = \mathbf{b}$ with a nonsymmetric matrix A to the solution of the matrix equation $H(\mathbf{r},\mathbf{x})^T = (\mathbf{b},\mathbf{0})^T$ where $H = \begin{bmatrix} O & A \\ A^T & I \end{bmatrix}$ or $H = \begin{bmatrix} O & A \\ A^T & O \end{bmatrix}$. Here and hereafter I, W^T,$\mathbf{v}^T$,O and $\oplus$ denote the identity matrix, the transpose of a matrix W and of a vector $\mathbf{v}$, the null matrix and the null vector, respectively. On the other hand, it can be shown that the algorithm that we will suggest can be extended to the case of nonsymmetric systems $A\mathbf{x} = \mathbf{b}$ also, as long as a family of $s(m+n)$-separators is known for the associated graph of A.

Another apparent difficulty is that the dioid elements may have no inverses regarding the operations $\oplus$ and $\ast$. That difficulty is, however, resolved due to the generalized Jordan elimination algorithm, which requires only the operations $\oplus$ and $\ast$, see [GM], *Section* 3.4.3. (The algorithm works unless generalized inverses of some computed values do not exist but this is a relatively weak restriction.) We combine that algorithm with computing a recursive factorization of Section 4 of [PR] for the matrix $A_0 = PAP^T$ where P is a permutation matrix obtained in the process of computing that recursive factorization. In [PR] the recursive factorization of A_0 is defined by the following matrix and block-matrix equations where h=0,1,...,d-1, d=O(log n),

$$A_h = \begin{bmatrix} X_h & Y_h^T \\ Y_h & Z_h \end{bmatrix}, \quad Z_h = A_{h+1} + Y_h X_h^{-1} Y_h^T,$$

$$A_h = \begin{bmatrix} I & 0 \\ Y_h X_h^{-1} & I \end{bmatrix} \begin{bmatrix} X_h & 0 \\ 0 & A_{h+1} \end{bmatrix} \begin{bmatrix} I & X_h^{-1} Y_h^T \\ 0 & I \end{bmatrix},$$

$$A_h^{-1} = \begin{bmatrix} I & -X_h^{-1} Y_h^T \\ 0 & I \end{bmatrix} \begin{bmatrix} X_h^{-1} & 0 \\ 0 & A_{h+1}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -Y_h X_h^{-1} & I \end{bmatrix}.$$

Computing over dioids we should replace the inverse W^{-1} of every matrix W with its generalized inverse W^* and either dispense with the signs - or replace them with $\oplus$. Then we would arrive at the following recursive factorization that we will substantiate below.

$$A_h = \begin{bmatrix} X_h & Y_h^T \\ Y_h & X_h \end{bmatrix}, A_{h+1} = Z_h \oplus Y_h X_h^* Y_h^T, \tag{2}$$

$$A_h = \begin{bmatrix} I & O \\ Y_h X_h^* & I \end{bmatrix} \begin{bmatrix} X_h & O \\ O & A_{h+1} \end{bmatrix} \begin{bmatrix} I & X_h^* Y_h^T \\ O & I \end{bmatrix},$$

$$A_h^* = \begin{bmatrix} I & X_h^* Y_h^T \\ O & I \end{bmatrix} \begin{bmatrix} X_h^* & O \\ O & A_{h+1}^* \end{bmatrix} \begin{bmatrix} I & O \\ Y_h X_h^* & I \end{bmatrix}, \tag{3}$$

h=0,1,...,d-1. We should verify that (2) and (3) indeed define the generalized inverse of A over the given dioid. To do that, we apply the generalized Jordan algorithm to the 2×2 block matrices A_h of (2), which have decreasing sizes as h grows from 0 to d-1. The proof of Theorem 4.3.6, [GM], pp. 108-110, and consequently of the equations (4.3.1) and of the subsequent equations on p. 16 of [GM] are easily extended to the case of block-matrices. These equations for the case N=2 immediately imply the validity of (2) and (3). (Expand (3) and adjust the notation of [GM], p. 110.)

It remains to estimate the number of operations in the dioid required in order to compute A^*b using the recursive factorization (2), (3). We proceed similarly to deriving the estimates for the recursive factorization in [PR], noting that for some auxiliary $s(\alpha^h n) \times s(\alpha^h n)$ block-matrices B, (where $\alpha < 1$ $h=k,k-1,\ldots,0$, $k=O(\log n)$), we need to compute B^*v,v being a fixed vector, B^* being defined by (1) where B^* and B substitute for A^* and A, respectively. It is easy to extend the assumed property that $A^{(q+1)} = A^{(q)}$ for $q \geq n-1$ to the equations $B^{(q+1)} = B^{(q)}$ for $q \geq s(\alpha^h n)-1$ for A and B are associated with the path problems of same kind, having only different sizes n and $s(\alpha^h n)$, respectively. Similarly to [PR], for the evaluation of B^* given B, we apply the cited earlier algorithms for the dense matrix case. These algorithms require $s^2(4s-1)(\lceil \log_2 s \rceil - 1)$ operations in the dioid where $s=s(\alpha^h n)$. Applying parallelization we arrive at the favorable complexity bounds of $O(\log n \log^2 s(n))$ parallel steps and $|E| + s^3(n)/\log s(n)$ processors for computing the recursive factorization of A^* and $O(\log n \log s(n))$ parallel steps and $|E| + s^2(n)$ processors for computing A^*b for every b where the recursive factorization of A^* is available. Here $|E|$ denotes the number of edges of the graph associated with the matrix A. This gives algorithms for both single source path problems (where **b** is fixed) and all pair path problem (where we may just perform the evaluation for all the n coordinate vectors **b**). Comparison of the latter estimates with the estimates for the cost of the straightforward algorithms for computing A^* and A^***b**, which we recalled earlier, shows that our extension of the nested dissection algorithm of [PR] substantially accelerates the solution of *both* single source and all pair path problem.

The authors thank Sally Goodall for typing this paper.

References

[GM] M. Gondran and M. Minoux 1984, *Graphs and Algorithms*, Wiley-Interscience, New York.

[PR] V. Pan and J. Reif 1984, Fast and Efficient Solution of Linear Systems, Tech. Report TR-02-85, *Center for Research in Computer Technology, Aiken Computation Laboratory, Harvard Univ.*, Cambridge, Mass., (extended abstract in *Proc. 17-th Ann. ACM STOC*, 143-152, Providence, R.I.).

[PRa] V. Pan and J. Reif 1985, Fast and Efficient Algorithms for Linear Programming and for the Linear Least Squares Problem, Techn. Report TR-11-85, *Center for Research in Computer Technology, Aiken Computation Laboratory, Harvard Univ.*, Cambridge, Mass.

[T81] R.E. Tarjan 1981, Fast Algorithms for Solving Path Problems, *J. ACM* 28,3, 594-614.

END

FILMED

1-86

DTIC